Composable, Declarative, Reproducible, Verifiable Network and Service Configurations

Jürgen Schönwälder, Constructor University

October 16, 2024

1 Introduction

Looking back at the Network Management Workshop more than 20 years ago [1], it is revealing how much the recommendations were shaped by substantial discussions in the protocol developer and network operator communities at that time. Although some hot discussion topics from that time have since become largely irrelevant, certain technology choices suggested back then still have an impact on our network management technology and standards today (albeit not always in the most optimal manner).

The IETF has undoubtedly made significant advancements subsequent to the workshop. The YANG modeling language [2] (originally published in 2010) has largely replaced the SMIv2 [3] language used to write MIB modules for SNMP [4]. There are meanwhile more than 80 standards-track RFCs with YANG modules, and the number is growing. Other standards developing organizations like the IEEE have adopted YANG for their purposes. Similarly, NETCONF [5] (originally published in 2006) and RESTCONF [6] (published in 2017) have seen widespread adoption.

While the YANG data modeling language has been successful, it is not without flaws and limitations. It is thus desirable that YANG technology continues to evolve to address some of the shortcomings. However, a more significant step forward is required if we think about a new *era* of network management operations, similar to the major leap we made after the IAB workshop more than 20 years ago.

This position statement is based on the recognition that most of the work done within the IETF after the IAB workshop in 2002 focused on making device configuration more robust and to support configuration automation at the device interface. While this provided us with a valuable foundation for driving automation today, there is still a gap of open standards when we consider the configuration of a network as a whole. In the following, first some remarks are made on the evolution of existing technology in Section 2 before discussing aspects to consider once the IETF shifts its focus towards network and service configuration in the future in Section 3. Section 4 concludes this position paper with a short summary and two recommendations.

2 Evolution of Existing Technology

The IETF has a strong track record of maintaining and evolving technology. A careful evolution of technology is important to keep it relevant and to react to requirements surfacing from the users of a technology. This applies to both the data modeling language YANG and the communication protocols, primarily still NETCONF and RESTCONF as far as the IETF is concerned. In the following, we will more specifically focus on the data modeling aspects.

For the YANG data modeling language, there is a collection of requests to improve the language and to fix known limitations on the YANG next issue tracker. It is good to see that the IETF is starting to discuss these requests and hopefully this eventually leads to an improved version of

YANG. There is also good work underway in the IETF to add a package mechanism to the YANG language, which will help manage growing collections of modules.

Additional work could increase the declarative nature of the language and to improve the mechanisms available to express constraints on valid configurations. For example, the current mechanisms, like XPATH [8] in `when` and `must` statements, or regular expressions to restrict the value space of data types, have their complexities and limitations. Experience tells us that it is difficult to get XPATH expressions in `when` and `must` right and there are limitations on their expressive power, some are addressed by additional function libraries.

Similarly, regular expressions used to constrain value spaces of data types often turn out to be difficult. Authors trying to capture ABNF [9] grammars in regular expressions usually experience these limitations. One could investigate other mechanism to constrain value spaces that can deal with more complex formats.

Another aspect is to improve support for building data models from reusable model components. YANG is somewhat limited in this regard, as it solely provides groupings as reusable structures, which have their own constraints and limitations. It may be helpful to draw inspiration from modern type systems, such as those foound functional programming languages, which are effective in supporting composition.

3 Focus on Network and Service Configuration

To make a leap step forward, it is necessary to shift the focus from device configuration to network and service configuration. So far, there has been limited success in being able to describe and exchange configurations of entire networks or complex services. While some attempts were made to model network services using YANG data models, it is not clear whether this is a good solution or whether additional technologies could better support describing network or service configurations.

3.1 Composable Configurations

To handle complexity, there is a need to better support composition of configurations. There seems to be a lack of open technology enabling operators to define reusable configuration components for specific services that do compose well into a larger network and service configurations. There is room for open technology that paves the way to dedicated tools that can validate or even verify desired properties of a composition.

3.2 Declarative Configurations

YANG helped to make configuration data models more declarative than they used to be before YANG. In addition, NETCONF introduced support for robust configuration change transactions involving multiple network elements. The Network Management Datastore Architecture (NMDA) [7] provides a detailed model of the configuration information flow in a system. Devices supporting NMDA can clearly expose the origin of configuration settings (e.g., intended, system, learned, default). NMDA also allows to distinguish clearly between intended and applied configuration.

Building on this, it is desirable to expand things to the network scope. A network configuration model should be declarative and at the same time detailed enough that it is possible to verify that certain goals have been met. This includes mechanisms that can be used by network operators to express their specific deployment constraints that go beyond rather generic constraints such as connectivity. Certain network level constraints may even translate to constraints that must be met by device configurations. Note that YANG currently has no good support for adding deployment specific constraints to data models. It can be expected that tackling the declarative network and service configurations may lead to new requirements for the YANG language.

3.3 Reproducible Configurations

Network configurations are constantly evolving and there is a need to support a smooth transition between network configurations. We currently have no standardized mechanisms to express even simple things like "*bring all systems back to network configuration X*" that worked yesterday.

Some orchestrators may provide such functionality by translating a network wide version change to a number of device specific configuration change transactions that are pushed to network elements. However, moving between network configurations could be much faster and likely also be more robust and efficient if devices would maintain multiple named and versioned configurations with fast transitions between them.

In the computing world, modern operating systems like NixOS [10] explore radically new ways to think about the configuration of an entire operating system. NixOS is based on a declarative software package management system. Package configurations are associated with stable identifiers that also capture versions of the dependencies. This makes NixOS configurations reproducible and switching between configurations is fast, easy, and robust. We do not seem to have something equivalent in the networking space.

3.4 Verifiable Configurations

The ultimate goal of declarative and reproducible configurations are verifiable configurations that provably have certain properties such as resilience against certain types of failures or robustness against certain attacks. This requires to establish layered formal models that can interface to proof assistants, such as Isabelle [11], Coq [12], or Lean4 [13].

For example, formally verifying that a configuration of a complex network enforces certain access control policies is crucial once several firewalls and tunnels are involved. Implementing security policies correctly has been shown to be challenging, in particular in enterprise networks that often grow in somewhat uncontrolled fashion. Similarly, verifying that the configuration of a complex network supports suitable backup paths that can keep core services connected in situations of component failures or certain attacks is invaluable.

The availability of reliable topology information is a crucial prerequisite for reasoning about network configurations. While some work has already been done in this space, it remains unclear whether the existing models are sufficient to drive network-wide configuration verification systems.

4 Conclusions

Over twenty years ago, the IAB Network Management workshop laid the foundation for robust and programmable device configuration management interfaces that are widely deployed today. The work accomplished by the IETF after the workshop has lead to notable advances when it comes to automated and robust device configurations. This leads to the first recommendation:

R01 The IETF should actively maintain and evolves the technology enabling robust device configuration. It is necessary to address shortcomings and limitations in a timely manner.

Today's IAB workshop on the *Next Era* of Network Management Operations could signal the beginning of a shift of the focus towards network configurations (as opposed to just device configurations). This is a logical progression towards managing networks and services instead of collections of devices. This leads to the second recommendation:

R02 The IETF and the IRTF should develop technologies that support composable, declarative, reproducible, and verifiable network and service configuration. Ideally, the technologies should create an ecosystem of interoperable tools supporting tasks throughout the network and service life-cycle.

References

[1] J. Schönwälder. Overview of the 2002 IAB Network Management Workshop. RFC 3535, International University Bremen, May 2003.

[2] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020, Tail-f Systems, October 2010.

[3] K. McCloghrie, D. Perkins, and J. Schönwälder. Structure of Management Information Version 2 (SMIv2). RFC 2578, Cisco Systems, SNMPinfo, TU Braunschweig, April 1999.

[4] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and Applicability Statements for Internet Standard Management Framework. RFC 3410, SNMP Research, Network Associates Laboratories, Ericsson, December 2002.

[5] R. Enns. NETCONF Configuration Protocol. RFC 4741, Juniper Networks, December 2006.

[6] A. Bierman, M. Björklund, and K. Watsen. RESTCONF Protocol. RFC 8040, YumaWorks, Tail-f Systems, Juniper Networks, January 2017.

[7] M. Björklund, J. Schönwälder, P. Shafer, K. Watsen, and R. Wilton. Network Management Datastore Architecture (NMDA). RFC 8342, Tail-f Systems, Jacobs University, Juniper Networks, Cisco Systems, March 2018.

[8] J. Clark and S. DeRose. XML Path Language (XPath) Version 1.0. Recommendation, W3C, November 1999.

[9] D. Crocker and P. Overell. Augmented BNF for Syntax Specifications: ABNF. RFC 4234, Brandenburg InternetWorking, THUS plc., October 2005.

[10] Eelco Dolstra and Andres Löh. Nixos: a purely functional linux distribution. *SIGPLAN Notices*, 43(9):367–378, September 2008.

[11] T. Nipkow, L.C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer, 2024.

[12] Adam Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to the Coq Proof Assistant*. The MIT Press, 2013.

[13] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28: 28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings*, page 625–635, Berlin, Heidelberg, 2021. Springer-Verlag.